

H 6 — 2 1 4 6 9 0

Purpose: To enable synchronization control of a multimedia object in detail.

Constitution: A state 1 in a basic class A is constituted of states of state numbers 1.0, 1.1, and 1.2. Among them, a state number (n) of the basic class A is selected, and further subdivided into a plurality of classes. The subdivided classes are located in a derived class B of the basic class A. For instance, the derived class B is constituted of the states of the state numbers (n.0), (n.1), and (n.m). Similarly, regarding one of the state numbers, for example, the state number (n.m), a derived class C of the derived class B is indicated by a set of states including state numbers (n.m.0), (n.m.1), and (n.m.k). In such a manner, the classes are hierarchically defined.

(19)日本国特許庁(JP)

(12) 公開特許公報(A)

(11)特許出願公開番号

特開平6-214690

(43)公開日 平成6年(1994)8月5日

(51)Int.Cl. ⁵	識別記号	庁内整理番号	FI	技術表示箇所
G 0 6 F 3/00		Z 7165-5B		
9/44	3 3 0	Z 9193-5B		

審査請求 未請求 請求項の数5 FD (全 14 頁)

(21)出願番号 特願平5-21728

(22)出願日 平成5年(1993)1月14日

(71)出願人 000002185

ソニー株式会社

東京都品川区北品川6丁目7番35号

(72)発明者 山岸 靖明

東京都品川区北品川6丁目7番35号 ソニー株式会社内

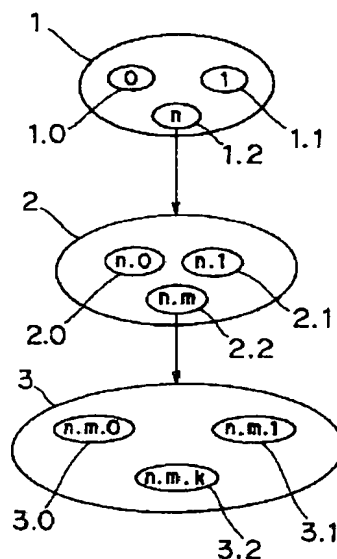
(74)代理人 弁理士 稲本 義雄

(54)【発明の名称】 情報同期制御方法、情報伝送装置および情報出力装置

(57)【要約】

【目的】 マルチメディアオブジェクトのより細かな同期制御を可能とする。

【構成】 基本クラスAにおける状態1は、状態番号1. 0, 1. 1および1. 2の状態により構成される。このうち、例えば基本クラスAの導出クラスBにおける基本クラスAの状態番号nの状態を細分化した状態の集合は、状態番号n. 0, n. 1, n. mの状態により構成される。同様に、導出クラスBの導出クラスCは、導出クラスBの状態番号n. mの状態を細分化した状態の集合3で表され、この集合3は、状態番号n. m. 0, n. m. 1, n. m. 2およびm. m. kの状態により構成される。このように、クラスが階層的に定義される。



基本クラスA
class A{
state:0,1,2.....
...
n:状態(ア);
...
}

導出クラスB
class B:A{
state:n.0,n.1.....
...
n.m:状態(イ);
...
}

導出クラスC
class C:B{
state:n.m.0,n.m.1,
n.m.2,.....
...
n.m.k:状態(ウ);
...
}

クラスの階層的定義による状態遷移の継承

【特許請求の範囲】

【請求項1】 複数のメディアからの情報を出力する際の各情報間の同期関係を制御する情報同期制御方法において、

各情報のメディアからの出力状態と、前記出力状態の遷移を階層的に定義し、前記出力状態とその遷移を利用したスクリプトをもとに、各情報の出力状態の同期を制御することを特徴とする情報同期制御方法。

【請求項2】 前記各情報のメディアからの出力状態と、その遷移は、基本クラス、前記基本クラスの下位の導出クラス、または他の導出クラスの下位の導出クラスにより定義されることを特徴とする請求項1に記載の情報同期制御方法。

【請求項3】 前記基本クラスと導出クラスは、前記メディアの状態を定義するステートと、前記状態の遷移を表わすイベントにより定義されることを特徴とする請求項2に記載の情報同期制御方法。

【請求項4】 情報のメディアからの出力状態とその遷移を階層的に定義するとともに、前記出力状態とその遷移を利用したスクリプトを作成する第1の作成手段と、前記メディアのオブジェクトデータを作成する第2の作成手段と、前記第1の作成手段と第2の作成手段の出力を合成して伝送する伝送手段とを備えることを特徴とする情報伝送装置。

【請求項5】 入力信号から、階層的に定義された情報のメディアからの出力状態とその遷移を利用したスクリプトと、メディアのオブジェクトデータとを分離する分離手段と、前記分離手段により分離されたスクリプトを解釈する解釈手段と、前記解釈手段の出力に対応して、前記分離手段により分離された前記オブジェクトデータを処理する処理手段とを備えることを特徴とする情報出力装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、マルチメディア情報を表示／再生する情報処理装置において、情報要素間の同期関係を制御する情報同期制御方法、情報伝送装置および情報出力装置に関する。

【0002】

【従来の技術】ワークステーション、記憶装置、通信装置などの発展に伴い、テキストデータや静止画像ばかりでなく、サウンドシークエンス、オーディオビジュアル等の新しい表現メディア（相互交換されるデータの種別、符号化された形式で記述される情報そのもの）を利用したアプリケーションプログラムが普及しつつある。

【0003】この表現メディアの多様化により、各々のメディアを組み合わせた使い方、すなわち、マルチメディア（複数種類の表現メディア）を扱う要求が増加し、

それとともに、マルチメディア情報を異なるプラットフォーム間で共有するための標準的な情報の交換形式の規定が望まれている。

【0004】その1つの例が、情報要素（単一の表現メディアによって表される塊、以下、メディアオブジェクトとともに記す）間の表示／再生における同期制御方式（時間依存関係の制御方式）の規定である。

【0005】マルチメディア同期制御の制御方式には、大きく分けて、ビット／パケットレベル、オブジェクト（1つの意味のまとまり単位）レベルの2つの段階が考えられる。前者の例にはCCITT勧告H. 221等があり、後者の例にはISO/IEC/DIS10744 (HyTime) が挙げられる。

【0006】HyTime等、従来のオブジェクトレベルの同期制御方式においては、個々のメディアオブジェクトの出現位置／領域が、空間および時間軸上にアドレスリングされている。つまり、表示／再生位置が静的にスケジュールされている。

【0007】

【発明が解決しようとする課題】マルチメディア情報を表示／再生する際の情報要素間の同期関係を制御する方式のうち、オブジェクトレベルの同期制御方式においては、表示／再生位置を静的にスケジュールする方法が一般的である。

【0008】しかしながら、このような静的なスケジュールリングのみをサポートする制御方式は、ユーザからの対話のタイミングによる同期を制御する能力が貧弱である課題を有している。

【0009】例えば、いま、ここに、マルチメディア情報表示／再生装置があり、この装置は同期関係を記述したスクリプト（情報要素間の表示／再生制御のシナリオを記述したプログラム）を読み込み、それを実行することが可能であるものとする。そして、そのスクリプトには、ディスプレイ装置の画面上にボタンと動画表示領域を配置し、ユーザが画面上のボタンを押すと、動画の再生が行なわれるというように記述されているものとする。

【0010】ただし、そのスクリプトは、静的なスケジュールリングのみをサポートする同期制御記述方式で記述されている。ここでいう、静的にスケジュールリングするとは、“ボタンが押された場合には、動画をそのデータの最初から再生する。”というような記述の仕方のことをいう。

【0011】従って、この場合、動画の再生途中でボタンを押したとき、動画が再生途中であるか否かにかかわらず、動画の最初から再生が行なわれ、再生をそのまま続けるか、最初から再生し直すかの選択ができない。つまり、ボタンを押したときの動画の表示状態により、再生の仕方を細かく制御することができないという問題点がある。

3

【0012】従って、ユーザとの対話管理を含めた、マルチメディアオブジェクトの同期制御を効果的に記述するためには、スクリプトに、現在表示中のメディアオブジェクトの表示の状態に依存した同期制御を記述する能力（動的なスケジューリングの能力）が求められる。

【0013】さらに、メディアが異なると、その状態の種類が異なるため、すべてのメディアに共通な状態というものを定義することができるとともに、個々のメディアの特性に依存する状態を定義することができる能力を有することが望ましい。

【0014】例えば、静止画と動画を例にとると、静止画は時間に依存しないデータ（離散メディア）であるため、それがもつ表示の状態は大きく分けると、“表示中”、または“表示されていない”、という2つの状態のいずれかである。これに対して、動画の場合は、それが時間に依存するデータ（連続メディア）であるため、“表示中”、または“表示されていない”、という2つの状態に区分される他、“表示中”はさらに、“通常の速度での再生”と、“倍速再生”に区分される。

【0015】本発明は、以上のような問題点に鑑みてなされたものであり、マルチメディア情報の同期制御を行なうに際し、マルチメディアオブジェクトの表示中のマルチメディア情報要素の状態に依存した細かな同期制御を可能とすることを目的としている。

【0016】

【課題を解決するための手段】請求項1に記載の情報同期制御方法は、複数のメディアからの情報を出力する際の各情報間の同期関係を制御する情報同期制御方法において、各情報のメディアからの出力状態と、出力状態の遷移を階層的に定義し、出力状態とその遷移を利用したスクリプトをもとに、各情報の出力状態の同期を制御することを特徴とする。

【0017】各情報のメディアからの出力状態と、その遷移は、基本クラス、基本クラスの下位の導出クラス、または他の導出クラスの下位の導出クラスにより定義することができる。

【0018】また、基本クラスと導出クラスは、メディア

基本クラスのシンタクスは、以下のように定義される。

```
class <クラス名(文字列)> {
    state:
        [<状態番号(正の整数)> : <状態名(文字列)>]*
    event:
        [<イベント番号(正の整数)> : <イベント名(文字列)> :
        <状態番号(正の整数)> → <状態番号(正の整数)>]*
}
```

【0024】“state”の項には、状態番号と対応する状態名の組を記述する。状態番号は、この“state”の項内で一意な整数である。“event”の項には、“<状態番号(正の整数)> → <状態番号(正の整数)>”で示される<状態番号(正の整数)> → <状

4

態の状態を定義するステートと、状態の遷移を表わすイベントにより定義することができる。

【0019】請求項4に記載の情報伝送装置は、情報のメディアからの出力状態とその遷移を階層的に定義するとともに、出力状態とその遷移を利用したスクリプトを作成する第1の作成手段としてのクラス定義スクリプト作成部11と、メディアのオブジェクトデータを作成する第2の作成手段としてのメディアオブジェクトデータ作成部12と、クラス定義スクリプト作成部11とメディアオブジェクトデータ作成部12の出力を合成して伝送する伝送手段としての転送制御部13とを備えることを特徴とする。

【0020】請求項5に記載の情報出力装置は、入力信号から、階層的に定義された情報のメディアからの出力状態とその遷移を利用したスクリプトと、メディアのオブジェクトデータとを分離する分離手段としての転送制御部16と、転送制御部16により分離されたスクリプトを解釈する解釈手段としてのスクリプト解釈実行部17と、スクリプト解釈実行部17の出力に対応して、転送制御部16により分離されたオブジェクトデータを処理する処理手段としての表示制御部18とを備えることを特徴とする。

【0021】

【作用】本発明の情報同期制御方法、情報伝送装置および情報出力装置においては、各情報のメディアからの出力状態と、出力状態の遷移を階層的に定義し、出力状態とその遷移を利用したスクリプトをもとに、各情報の出力状態の同期が制御される。従って、マルチメディアオブジェクトのより細かな同期制御が可能となる。

【0022】

【実施例】以下に実施例について説明する。本実施例においては、メディアがクラスとして定義され、クラスにおいて、メディアの状態遷移がイベントとして定義される。クラスは、さらに基本クラスと導出クラスとに区分される。

【0023】

状態番号(正の整数) > の状態遷移を引き起こすイベントの名前とイベント番号が組にして列挙される。イベント番号は、この“event”の項内で一意な整数である。なお、“[...]*”は、... が1つ以上出現することを表す。なお、後述するように、状態番号”

0”として設定された状態は、初期状態とされる。

定義される。

【0025】導出クラスのシンタクスは、以下のように

```
class <クラス名(文字列)> : <基本クラス名または導出クラス名(文字列)>
{
    state:
        [<状態番号(正の整数列)> : <状態名(文字列)>;]*
    event:
        [<イベント番号(正の整数)> : <イベント名(文字列)> :
            <状態番号(正の整数列)> → <状態番号(正の整数列)>;]*
}
```

【0026】この”クラス名”で識別される導出クラスは、親クラスである”基本クラス名”または”導出クラス名”で識別されるクラスの状態遷移と、それを引き起こすイベントを継承する。この場合、”状態番号”のフォーマットは、”.”で区切られた2つ以上の正の整数列で表わされ、最も右側の整数の左に隣接する整数は、1つ上位の親クラスの状態番号を表わす（後述の例を参照のこと）。すなわち、導出クラスの記述は、親のクラス（基本クラスまたは導出クラス）のある状態を、さらに細分化した状態を定義することになる。

【0027】例えば、ある基本クラスにおいて、状態番号n（nは正の整数）が定義される場合、その導出クラス（子クラス）において、親クラスの状態番号nに対応する状態をさらに細分化して定義するときは、状態番号をn.m（n、mは正の整数）とする。

【0028】さらに、その導出クラス（子クラス）の導出クラス（孫クラス）において、子クラスの状態番号n.mに対応する状態をさらに細分化して定義する場合は、状態番号をn.m.k（n、m、kは正の整数）とする。

【0029】以上のことを、図1を参照してさらに説明する。図1において、1は、基本クラスAにおける状態の集合を表し、1.0は状態番号0の状態を表し、1.

<オブジェクト定義部> =

[<オブジェクト変数> = new <クラス名>(<内容部識別子>);]*

【0032】オブジェクト定義部は、このスクリプトで記述の対象となるオブジェクトを定義する部分である。オブジェクト変数は、指定されたクラスのインスタンスを特定する識別子となる。

【0033】内容識別子は、次式で表わされるように、そのオブジェクトの表現メディアそのもののデータを特定する識別子である。

<内容識別子> =

content_identifier = <正の整数>

【0034】イニシャルメッセージパッシング記述部は、次式で表わされ、そのスクリプトが起動されたとき、一番最初に実行される部分である。

<イニシャルメッセージパッシング記述部> =

<イベントメッセージパッシング記述部> =

[switch (<イベント判定部>) {

1は状態番号1の状態を表し、1.2は状態番号nの状態を表している。また、2は、基本クラスAの導出クラスBにおける、Aの状態番号nの状態1.2を細分化した状態の集合を表し、2.0は状態番号n.0の状態を表し、2.1は状態番号n.1の状態を表し、2.2は状態番号n.mの状態を表している。同様に、3は、導出クラスBの導出クラスCにおける、Bの状態番号n.mの状態2.2を細分化した状態の集合を表し、3.0は状態番号n.m.0の状態を表し、3.1は状態番号n.m.1の状態を表し、3.2は状態番号n.m.kの状態を表している。

【0030】上述のクラス定義に記述された状態遷移に従うオブジェクトの宣言と、それらの表示制御手続きがスクリプトに記述される。スクリプトのシンタクスは以下のようなされる。

script <スクリプト名> {

<オブジェクト定義部>

<イニシャルメッセージパッシング記述部>

<イベントメッセージパッシング記述部>

}

【0031】オブジェクト定義部は、次のように規定される。

<メッセージパッシング記述部>

【0035】メッセージパッシング記述部は次式で表わされる。

<メッセージパッシング記述部> =

do {[<イベント名> → <オブジェクト変数>;]*}

【0036】この式により、オブジェクト変数で特定されるオブジェクトに対してイベント名で特定されるイベントが送られる。その結果、対象オブジェクトの状態遷移が起こる。このイベント名は、対象のオブジェクトのクラス定義において定義されていなければならない。

【0037】イベントメッセージパッシング記述部は次式で表わされる。

[case (<状態判定部>) <メッセージパッシング記述部>]*
]]*

【0038】イベント転送の検出（後述）が起こったとき、すべてのswitch文のイベント判定部に指定されている条件のテストが行なわれ、それが成立した場合、case文の状態判定部に指定されている条件のテストが行なわれる。テストした条件が成立した場合、メッセージパッシング記述部に記述されているメッセージパッシングが行なわれる。

【0039】case文の状態判定部はなくてもよく、その場合には、ただちにメッセージパッシングが行なわれる。

【0040】イベント判定部は次式で表わされる。

<イベント判定部> =

<オブジェクト変数> . <イベント判定メソッド>

【0041】オブジェクト変数で特定される対象オブジェクトにおいて当該イベントが起こったか否かをテストする。起こっていれば、このイベント判定部が成立することとなる。

【0042】イベント判定メソッドは次式で表わされる。

<イベント判定メソッド> =

isSelected(), または、getMessage(<イベント名>)

【0043】isSelected() メソッドは、対象オブジェクトに対して”ユーザの選択”（後述）が起こったか否かをテストするものであり、getMessage() メソッドは、引き数<イベント名>で特定されるイベントが対象オブジェクトにおいて起こったか否かをテストするものである。

【0044】状態判定部は次式で表わされる。

<状態判定部> =

<オブジェクト変数> . <状態判定メソッド>

[<論理演算子> <オブジェクト変数> . <状態判定メソッド>]*

【0045】これは、オブジェクト変数で特定される対象オブジェクトにおける状態をテストするものであり、対象オブジェクトが下記の状態判定メソッドの引き数<状態番号>で特定される状態にあるか否かをテストする。

<状態判定メソッド> =

testState(<状態番号>)

【0046】なお、<論理演算子> はC言語の論理演算子(&& (および), || (または))と同じ意味である。

【0047】上述のクラス定義の表記構文と、スクリプトの表記構文にもとづいて、具体的なクラス定義とスクリプトの例を送り側で記述し、それらを伝送媒体を介して受け側のマルチメディア情報表示/再生情報処理装置に伝送し、受け側において、送り側で意図したシナリオにもとづいて、マルチメディア情報を同期制御する方法

を以下に説明する。

【0048】最初に、メディア(Media)クラスを定義する。このクラスに属するオブジェクトのとりえる状態として、例えば、次の2つの状態を定義する。

あ) 表示待機中(WAIT)

い) 表示中(RUN)

【0049】このオブジェクトが生成されたときの初期状態は”表示待機中(WAIT)”とする。

【0050】また、イベントとして、次の2つを定義する。

ア) run

イ) stop

【0051】以上のメディアの定義をまとめると、次のようになる。

```
class Media {
  state:
    0: WAIT;
    1: RUN;
  event:
    0:run: 0→1;
    1:stop: 1→0;
}
```

【0052】図2に示すように、このオブジェクトが状態”WAIT”の時に、”run”というメッセージを受けると、このオブジェクトの状態が、”RUN”に遷移する。また、オブジェクトが状態”RUN”の時に、”stop”というメッセージを受けると、このオブジェクトの状態が”WAIT”に遷移する。この状態遷移を表にまとめると、表1に示すようになる。

【0053】

【表1】

〈Mediaクラスの状態遷移表〉

	イベント	
状態	run	stop
WAIT	RUN	—
RUN	—	WAIT

【0054】次に、上述のメディアクラスから導出される連続メディア(Continuous Media: 再生状態が時間に依存するメディア)クラスを定義す

る。この連続メディアクラスは上述のメディアクラスの導出クラスであるから、上記2つの状態”WAIT”と”RUN”を有する。

【0055】さらに、”RUN”の状態を

あ) 通常再生 (PLAY)

い) 早送り再生 (FAST_PLAY)

の2つの状態に分ける。このうち、”通常再生 (PLAY)”を初期状態とする(状態番号を0として設定した方が初期状態とされる)。

【0056】このようなContinuousMedia 10 aクラスのクラス定義は次のようになる。

```
class ContinuousMedia : Media {
  state:
    1.0: PLAY;
    1.1: FAST_PLAY;
  event:
    1.0:playFast: 1.0→1.1;
    1.1:playNormal: 1.1→1.0;
}
```

【0057】ここで、”:Media”は、”Cont 20 inuousMedia”クラスが導出される親クラス

〈ContinuousMediaクラスの状態遷移表〉

状態	イベント	
	playFast	playNormal
PLAY	FAST_PLAY	—
FAST_PLAY	—	PLAY

【0061】次に、上述の各クラス(MediaとContinuousMedia)を用いて、所定のオブジェクトの表示状態を制御するために、例えば、下記のよ

```
script sample {
  a = new Media(content_identifier = 1);
  b = new ContinuousMedia(content_identifier = 2);
  c = new ContinuousMedia(content_identifier = 3);
  do {run→a;}
  switch (a.getMessage(run)) {
    do {run→c;}
  }
  switch (a.isSelected()) {
    case ((b.testState(0)) && (c.testState(0))) do {run→b;}
    case ((b.testState(1.0)) && (c.testState(0))) do {playFast→b;
      run→c;}
  }
```

を示す。また、状態番号が”n.m (n, mは正の整数)”となっているのは、状態”n.m: . . .”が、親クラスの状態”n: . . .”に含まれることを意味する。すなわち、状態”1.0:PLAY”と”1.1:FAST_PLAY”は、ともに親クラス”Media”の状態”1:RUN”の中に含まれる。

【0058】ただし、この場合、状態”1.0:PLAY”が初期状態(状態番号が0となっているため)となる。つまり、状態”WAIT”のときにイベント”run”を受け取ると、状態”PLAY”に遷移する。また、状態”PLAY”または”FAST_PLAY”のときにイベント”stop”を受け取ると、状態”WAIT”に遷移する。このように、子のクラス(下位のクラス)を定義する場合に、親のクラス(上位のクラス)に定義されている状態遷移を継承させることができる。

【0059】また、以上のContinuousMediaクラスの状態遷移を図と表に表わすと、図3と表2に示すようになる。

【0060】

【表2】

うなスクリプト(スクリプト名”sample”)を定義する。

```

11      case ((b.testState(0)) && (c.testState(1.0))) do {run→b;
        playFast→c;}
    }
    switch (c.getMessage(stop)) {
        case ((a.testState(1) && b.testState(0)) do {stop→a;}
    }
}

```

【0062】即ち、この実施例においては、オブジェクトaがクラスMediaとして規定され、オブジェクトbとcがクラスContinuousMediaとして規定される。そして、それぞれはID (content__identifier) 1, 2または3として識別されるようになされている。

【0063】do {run→a;}は、オブジェクトaにイベントrunを転送することを意味している。

```

switch (a.isSelected()) {
    case ((b.testState(0)) && (c.testState(0))) do {run→b;}
    case ((b.testState(1.0)) && (c.testState(0))) do {playFast→b;
        run→c;}
    case ((b.testState(0)) && (c.testState(1.0))) do {run→b;
        playFast→c;}
}

```

は、オブジェクトaが選択されたとき、各caseの条件が充足されるか否かを順次判定するものである。この

```

case ((b.testState(0)) && (c.testState(0))) do {run→b;}

```

は、オブジェクトbの状態が0 (WAIT) であり、かつ、オブジェクトcの状態も0 (WAIT) であるとき、オブジェクトbに対して、runのイベントを転送することを意味している。

【0065】図4に、上述のクラス定義とスクリプトを交換するマルチメディア情報表示／再生装置 (情報伝送装置または情報出力装置) の構成例を示す。この実施例においては、図中左側が、クラス定義とスクリプトの作成を行う送り側31とされ、図中右側が、表示制御を行なう受け側32とされている。

【0066】上述したクラス定義部のファイルとスクリプトのファイルが、送り側31のクラス定義／スクリプト作成部11において作成される。さらに、スクリプトにおけるcontent__identifier1, 2, 3に対応する表現メディアのオブジェクトが、メディアオブジェクトデータ作成部12において作成される。例えば、content__identifier1のデータとして、符号化方式Aで符号化された静止画のデータが含まれているファイルが作成され、content__identifier2のデータとして、符号化方式Bで符号化された音声のデータが含まれているファイルが作成され、さらに、content__identifier3のデータとして、符号化方式Cで符号化された動画のデータが含まれているファイルが作成され

```

switch (a.getMessage(run)) {
    do {run→c;}
}

```

は、aというオブジェクトがrunというメッセージを受けたとき、cというオブジェクトにrunというイベントを転送することを意味している。

【0064】

うち、例えば

る。

【0067】この例においては、スクリプトから判るように、content__identifier1に対応するオブジェクトはMediaクラスのオブジェクトであり、content__identifier2, 3に対応するオブジェクトはContinuousMediaクラスのオブジェクトであり、符号化方式にそれぞれA, B, Cを選んでいるが、他の符号化方式を用いることができるのは勿論である。すなわち、スクリプトそのものは、表現メディアの符号化方式に依存しないため、受け側32の能力に合わせた符号化方式を選択することができる。

【0068】それぞれの表現メディアオブジェクトファイルのデータ構造には、content__identifierを記述するレコード、符号化方式を記述するレコード、表示属性を記述するレコード、および符号化されたデータが格納されるレコードがある。

【0069】クラス定義スクリプト作成部11により作成したクラス定義部とスクリプトのファイル、並びに、メディアオブジェクトデータ作成部12により作成されたメディアオブジェクトのファイルは、転送制御部13に入力され、合成される。転送制御部13より出力されたデータは、通信制御装置14において所定の方式で変調され、伝送媒体10に伝送される。

【0070】伝送媒体10と転送制御プロトコルには、例えば、EthernetとTCP/IPを用いることができる。勿論、その他の組合せを用いることも可能である。

【0071】受け側32においては、伝送媒体10を介して伝送されてきたデータを通信制御装置15が受信、復調し、これを転送制御部16に供給する。転送制御部16は、受け取ったファイルを分離し、クラス定義部とスクリプトのファイルをスクリプト解釈/実行部17に供給し、残りの表現メディアオブジェクトのファイルを表示制御部18に供給する。

【0072】スクリプト解釈/実行部17においてクラス定義またはスクリプトが解釈される以前に、送り側31と受け側32の間で（具体的には送り側31のクラス定義/スクリプト作成部11と、受け側32のスクリプト解釈/実行部17の間で）、各クラスの状態の表示上の意味について合意がなされている必要がある。例えば、この実施例では、Mediaクラスの状態WAITとRUNや、ContinuousMediaクラスの

状態PLAYとFAST_PLAYの違いについて合意がなされている必要がある。

【0073】本実施例においては、状態を識別したり、状態間の遷移を定義することは可能であるが、個々の状態の表示上の意味については定義されていない。従って、スクリプトの交換（伝送）に先立ち、予め各クラスの状態の意味について合意を行なうという過程が必要となるのである。

【0074】以上のような合意が予めなされている状態で、最初にスクリプト解釈/実行部17において、クラス定義部の解釈が行なわれ、次にスクリプトの<オブジェクト定義部>が解釈され、オブジェクトa、b、cの状態情報を記憶する領域が確保される。状態情報を記憶する領域には、オブジェクトa、b、cの各々の状態番号が記憶される。クラス定義から、オブジェクトaは1桁の状態番号を持ち、オブジェクトb、cは2桁の状態番号を持つことが判るので、表3に示すように、各々のオブジェクトに対して2つの整数のフィールド1、2を持つレコードが定義される。

【0075】

【表3】

〈オブジェクトの状態記憶領域〉

オブジェクト	a		b		c	
フィールド	1	2	1	2	1	2

【0076】スクリプト解釈/実行部17では、すべてのオブジェクトを初期状態にセットする。すなわちオブ

ジェクトa、b、cのいずれも状態”WAIT”になり、すべてのフィールドは”0”で初期化される（表4の状態no.1参照）。

【0077】

【表4】

〈オブジェクトの状態遷移の例1〉

オブジェクト	a		b		c	
フィールド	1	2	1	2	1	2
状態no. 1	0	0	0	0	0	0
状態no. 2	1	0	0	0	0	0
状態no. 3	1	0	0	0	1	0
状態no. 4	1	0	1	0	1	1
状態no. 5	1	0	0	0	0	0
状態no. 6	0	0	0	0	0	0

【0078】このとき、スクリプト解釈/実行部17において、スクリプトの<イニシャルメッセージパッシング記述部>が解釈され、”run”という名前のイベントがオブジェクトaに送られる。

【0079】オブジェクトaは、上述したように、次のような文により、Mediaクラスのオブジェクトと宣言されている。

a = new Media(content_identifier = 1);

このため、Mediaクラスの定義により、イベント”run”を受け取ると、状態が0から1、すなわち状態”WAIT”から状態”RUN”に遷移することになる。従って、オブジェクトaの状態情報を記憶する領域の第1フィールドには状態番号”1”が記憶される（表4における状態no.2参照）。

【0080】このときまた、スクリプト解釈/実行部17は、表示制御部18に対して、オブジェクトaの内容部を表示するよう要求する。これにより、表示制御部18は、content_identifier=1に対応する表現メディアオブジェクトのデータ（符号化方式Aにて符号化されている）を、内蔵する復号器（符号化

方式Aに対応する、例えばMR(modified read)復号器)により復号化し、内蔵するディスプレイに表示させる。

【0081】このように、あるイベントがあるオブジェクトに送られ、その対象オブジェクトの状態遷移が起こる事象を、“イベントの転送が検出される”という。スクリプト解釈/実行部17は、表示制御部18によりイベント転送が検出された場合、スクリプトの<イベントメッセージパッシング記述部>の<イベント判定部>を1つずつテストする。

```
【0082】いまの場合、最初の文
switch (a.getMessage(run)) {
do {run→c;}
}
```

の<イベント判定部>に該当するイベントが起こったことになるため、do以降の<メッセージパッシング記述部>

```
do {run→c;}
```

が実行される。すなわち、“run”という名前のイベントがオブジェクトcに送られる。

【0083】オブジェクトcは文
c = new ContinuousMedia(content_identifier = 3);
により、ContinuousMediaクラスのオブジェクトと宣言されているが、ContinuousMediaクラスの定義には、イベント“run”を受け取った場合の状態遷移の定義がなされていない。しかしながら、このContinuousMediaクラスの親クラスがMediaクラスであるので、Mediaクラスに定義されているイベント“run”を受け取った場合の状態遷移定義に従い、オブジェクトcの状態が状態“RUN”、すなわち状態“PLAY”になる。なぜなら、ContinuousMediaクラスの定義に

```
case ((b.testState(0)) && (c.testState(1.0))) do {playFast→c;}
```

に該当するので(オブジェクトbが状態“WAIT”であり、かつ、オブジェクトcが状態“PLAY”であるので)、

<メッセージパッシング記述部>

```
do {playFast→c;}
```

が実行され、“playFast”という名前のイベントがオブジェクトcに送られる。

【0088】これにより、オブジェクトcは状態が、“PLAY”から“FAST_PLAY”に遷移する。従って、オブジェクトcの状態情報を記憶する領域に、状態番号“1.1”が記憶される(表4の状態no. 4参照)。またこのとき、オブジェクトcの動画の再生速度を早くする指令が、スクリプト解釈/実行部17から表示制御部18に通知され、表示制御部18において、オブジェクトcの動画が高速再生される。

【0089】この後、オブジェクトcの再生が終了する(データの最後まで再生してしまう)と、オブジェクト

より、親クラスの状態“RUN”の初期状態が状態“PLAY”になると宣言されているためである。

【0084】そして、オブジェクトcの状態情報を記憶する領域の第1フィールドに状態番号1が、第2フィールドに状態番号0が記憶される。すなわち、状態番号“1.0”が記憶されることになる(表4の状態no. 3参照)。このとき、スクリプト解釈/実行部17は、表示制御部18に対して、オブジェクトcの内容部、すなわち、content_identifier=3に対応する表現メディアオブジェクトを表示するよう要求する。表示制御部18は、この要求に対応して、符号化方式Cの復号器により動画データを復号し、ディスプレイに表示させる。

【0085】ここで、オブジェクトcの動画の内容が再生されている途中で、ユーザがディスプレイ上で、例えばマウス等の表示制御部18の管理下の指示装置により、オブジェクトaを指示したとすると、オブジェクトcスクリプト解釈/実行部17にイベント転送の検出が通知(この場合のイベントは表示状態の遷移を起こさない特別なものである。つまり、オブジェクトaの表示領域内においてユーザの指示が起こったという事象のみが通知)され、スクリプトの<イベントメッセージパッシング記述部>の<イベント判定部>がテストされる。

```
【0086】このとき、このイベントが、
switch (a.isSelected()) {
}
```

に該当するイベントであることが判るため、このswitch文の中の各case文の<状態判定部>がテストされる。

30 【0087】いまの場合、

cに対して“stop”というイベントが送られ、オブジェクトcの状態が状態“WAIT”に遷移し、状態番号が“0”に設定される(表4の状態no. 5参照)。

【0090】この“stop”イベントの転送は、スクリプトそのものには記述されていない特別なもので、時間に依存する連続メディアの再生が終了したときに限り、表示制御部18が“stop”イベントの転送が起こったものとみなし、スクリプト解釈/実行部17に対してイベント転送の検出を通知する。

```
【0091】このとき、スクリプトの<イベントメッセージパッシング記述部>の<イベント判定部>がテストされ、このイベントが、
switch (c.getMessage(stop)) {
}
```

に該当するイベントであることが判るため、このswitch文の中の各case文の<状態判定部>がテストされる。

【0092】いまの場合、オブジェクトaが状態”PLAY”であり、かつ、オブジェクトbが状態”WAIT”

case ((a.testState(1) && b.testState(0)) do {stop→a;}

の<メッセージパッシング記述部>

do {stop→a;}

が実行され、オブジェクトaの表示がディスプレイから消えて、状態が状態”WAIT”に遷移する（表4の状態no. 6参照）。

【0093】上述の表4の状態no. 3において、オブジェクトcの再生が終了してしまった後に、ユーザがディスプレイ上でオブジェクトaを指示した場合の表示の流れは、表5に示すようになる。

【0094】

【表5】

〈オブジェクトの状態遷移の例2〉

オブジェクト	a		b		c	
フィールド	1	2	1	2	1	2
状態no. 3	1	0	0	0	1	0
状態no. 4	1	0	0	0	0	0
状態no. 5	0	0	0	0	0	0

【0095】オブジェクトcの再生が終了した場合、オブジェクトcに対して”stop”というイベントが送られたものとみなされ、オブジェクトcの状態が”WAIT”に遷移し、状態番号が”0”に設定される（表5の状態no. 4参照）。スクリプト解釈／実行部17にイベント転送の検出が通知され、オブジェクトaの内容がディスプレイから消えて、状態が”WAIT”に遷移する（表5の状態no. 5参照）。

【0096】従って、以後は、ユーザがディスプレイ上でマウス等の指示装置により、オブジェクトaを指示することができないため、オブジェクトの状態遷移の表4の場合と異なる状態の流れとなる。

【0097】以上の処理をフローチャートに表すと、図5および図6に示すようになる。即ち、図5に示すように、最初にステップS1において、クラス定義部の解釈を行い、次にステップS2において、スクリプトの解釈／実行を行う。

【0098】このステップS2におけるスクリプトの解釈／実行の処理の詳細は、図6に示されている。

T”であるから、

【0099】即ち、最初にステップS11において、オブジェクト定義部の解釈が実行される。次にステップS12に進み、ステップS11におけるオブジェクト定義部の解釈に対応して、オブジェクト状態領域の確保が実行される。さらにステップS13においては、ステップS12において確保したオブジェクト状態記憶領域が初期化される。

【0100】次にステップS14において、イニシャルメッセージパッシング記述部が実行され、そしてステップS15において、オブジェクト状態が更新される。

【0101】ステップS16においては、event_check_counterが1にセットされる。そして、次にステップS17に進み、event_check_counter番目（いまの場合、この値は、ステップS16において1に設定されている）のイベント判定部のテストが実行される。テストの結果が正しいとき、ステップS18に進み、state_check_counterに1が設定される。

【0102】次にステップS19に進み、state_check_counter番目（いまの場合、この値は、ステップS18において1に設定されている）の状態判定部のテストが実行される。このテストの結果が正しいとき、ステップS20に進み、state_check_counter番目のメッセージパッシング記述部が実行される。

【0103】ステップS20における処理の後、ステップS21に進み、state_check_counterの値が1だけインクリメントされる。ステップS19において、state_check_counter番目の状態判定部のテストの結果が誤りであると判定された場合においては、ステップS20の処理はスキップされ（そのメッセージパッシング記述部は実行されず）、ステップS21の処理が実行される。

【0104】次にステップS22に進み、state_check_counterの値が1つのswitch文における状態判定部の総数と等しいか否かが判定される。ステップS22における判定がNOである場合においては、ステップS19に戻り、それ以降の処理が繰り返される。ステップS22における判定がYESである場合においては、ステップS23に進み、event_check_counterを1だけインクリメントする。ステップS17において、event_check_counter番目のイベント判定部のテストの結果が誤りであると判定された場合においては、ステップS18乃至S22における処理がスキップされ、ステップS23の処理が実行される。

【0105】ステップS23の次にステップS24に進

み、ステップS23でインクリメントしたevent_check_counterの値がイベント判定部の総数と等しいか否かが判定される。ステップS24の判定がNOである場合においては、ステップS17に戻り、それ以降の処理が繰り返し実行される。ステップS24の判定がYESである場合においては、ステップS25に進み、イベント転送の検出待ちの状態となる。そして、イベント転送が検出されたとき、ステップS25からステップS15に戻り、それ以降の処理が繰り返し実行される。

【0106】以上においては、送り側31から受け側32に対して、伝送媒体10を介してクラス定義とスクリプトを送送するようにしたが、例えば図7に示すように、蓄積媒体20を介して送り側31から受け側32に対して、これらのデータを伝送することが可能である。この場合においては、図4の実施例における通信制御装置14、15が、蓄積媒体制御装置21、22に変更される。蓄積媒体制御装置21は、転送制御部13より供給されるデータを蓄積媒体20に記録し、蓄積媒体制御装置22は、蓄積媒体20に記録されたデータを再生し、転送制御部16に出力する。

【0107】以上においては、クラス定義／スクリプトの交換に先立ち、予め各クラスの状態の表示上の意味について、送り側と受け側の間で、合意がなされていなければならないことを述べたが、これは、例えば、標準化機関等において、クラス定義の登録手続きを規定し、クラス名により一意に参照可能なクラス定義と、そこで記述されている状態の表示上の意味を記述したドキュメントを登録することにより実現することができる。

【0108】このようにすれば、クラス定義の部分を広範囲の異なるシステムにおいて共有することができる。また、スクリプトの汎用性（さまざまなベンダーからのシステム／プラットフォーム上でのスクリプトの互換性）が保証されることとなる。

【0109】

【発明の効果】以上の如く、本発明の情報同期制御方法、情報伝送装置または情報出力装置によれば、各情報のメディアからの出力状態と、出力状態の遷移を階層的に定義し、出力状態とその遷移を利用したスクリプトをもとに、各情報の出力状態の同期を制御するようにしたので、マルチメディアオブジェクトのより細かな同期制御が可能となる。

【図面の簡単な説明】

10 【図1】クラスの階層的定義による状態遷移の承継を説明する図である。

【図2】メディアクラスの状態遷移を説明する図である。

【図3】連続メディアクラスの状態遷移を説明する図である。

【図4】本発明の情報伝送装置および情報出力装置の一実施例の構成を示すブロック図である。

【図5】図4の実施例の動作を説明するフローチャートである。

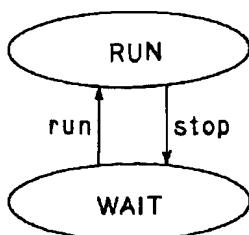
20 【図6】図5のステップS2のより詳細な処理を説明するフローチャートである。

【図7】本発明の情報伝送装置および情報出力装置の他の実施例の構成を示すブロック図である。

【符号の説明】

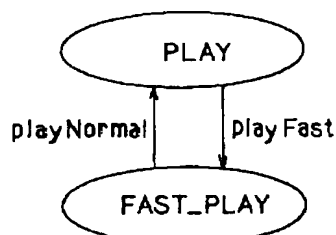
- 11 クラス定義スクリプト作成部
- 12 メディアオブジェクトデータ作成部
- 13 転送制御部
- 14, 15 通信制御装置
- 16 転送制御部
- 17 スクリプト解釈／実行部
- 18 表示制御部
- 20 蓄積媒体
- 21, 22 蓄積媒体制御装置

【図2】



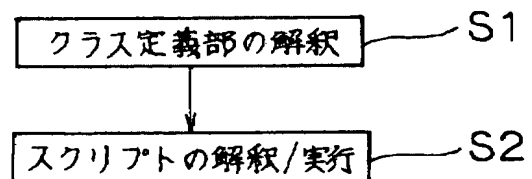
メディアクラスの状態遷移図

【図3】



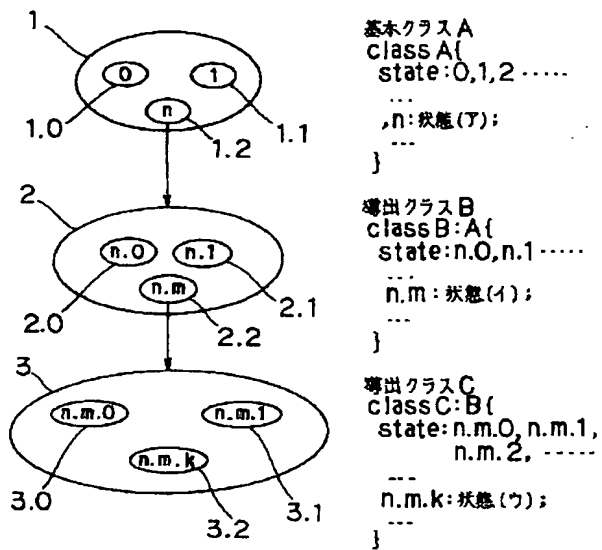
連続メディアクラスの状態遷移図

【図5】



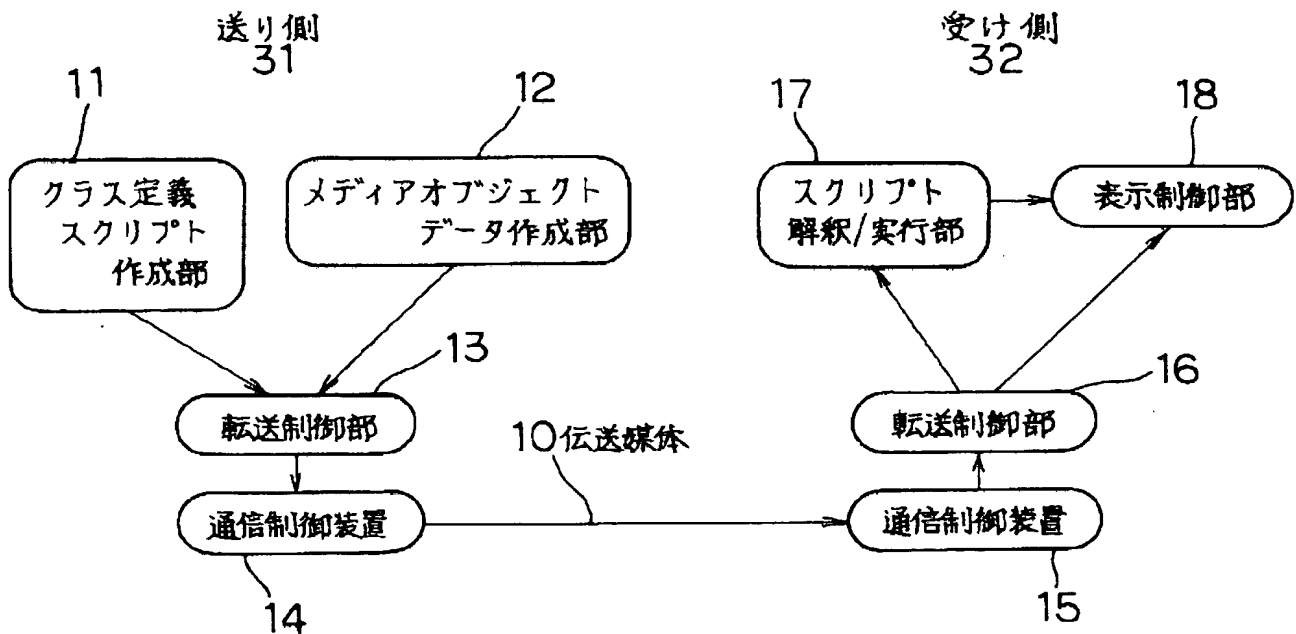
処理のアルゴリズム

【図1】



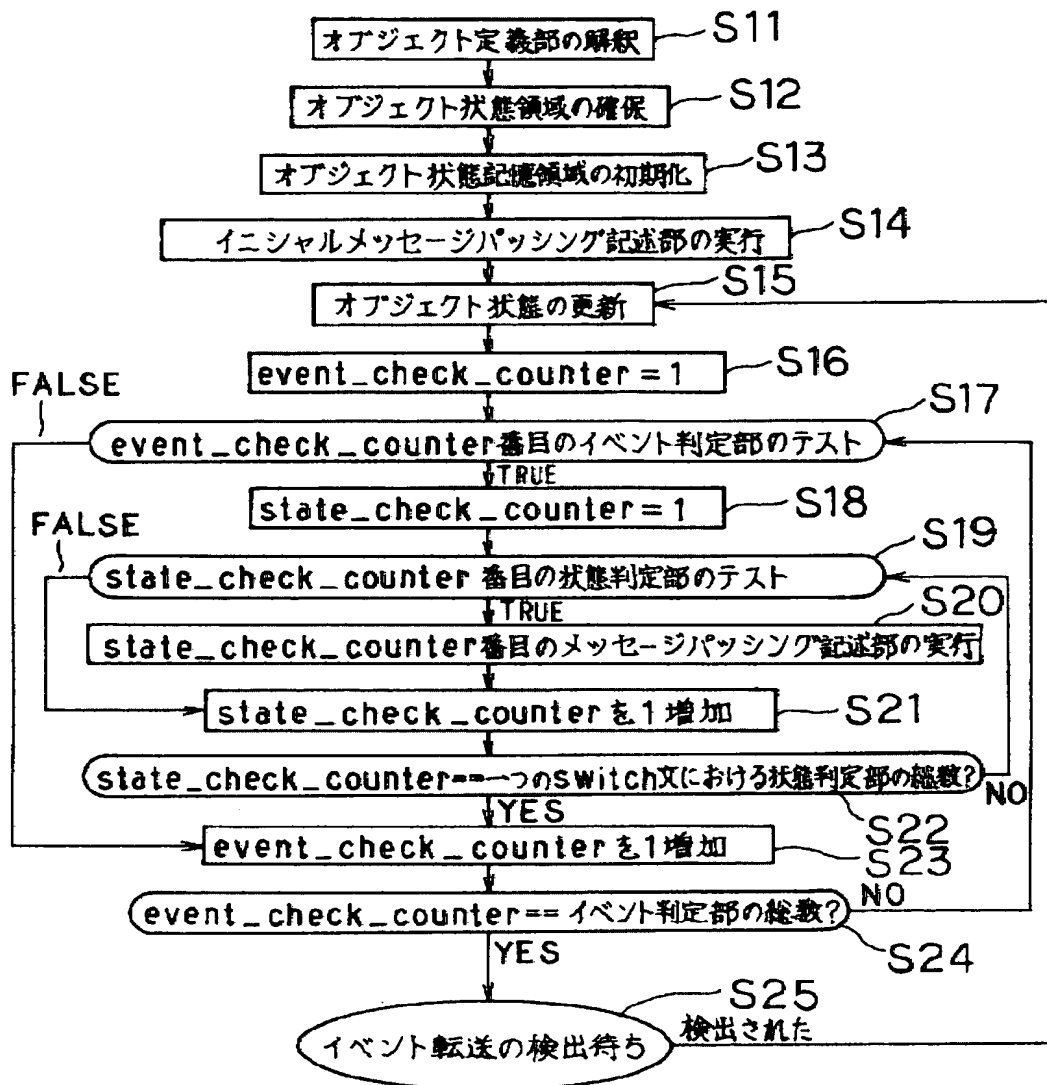
クラスの階層的定義による状態遷移の継承

【図4】



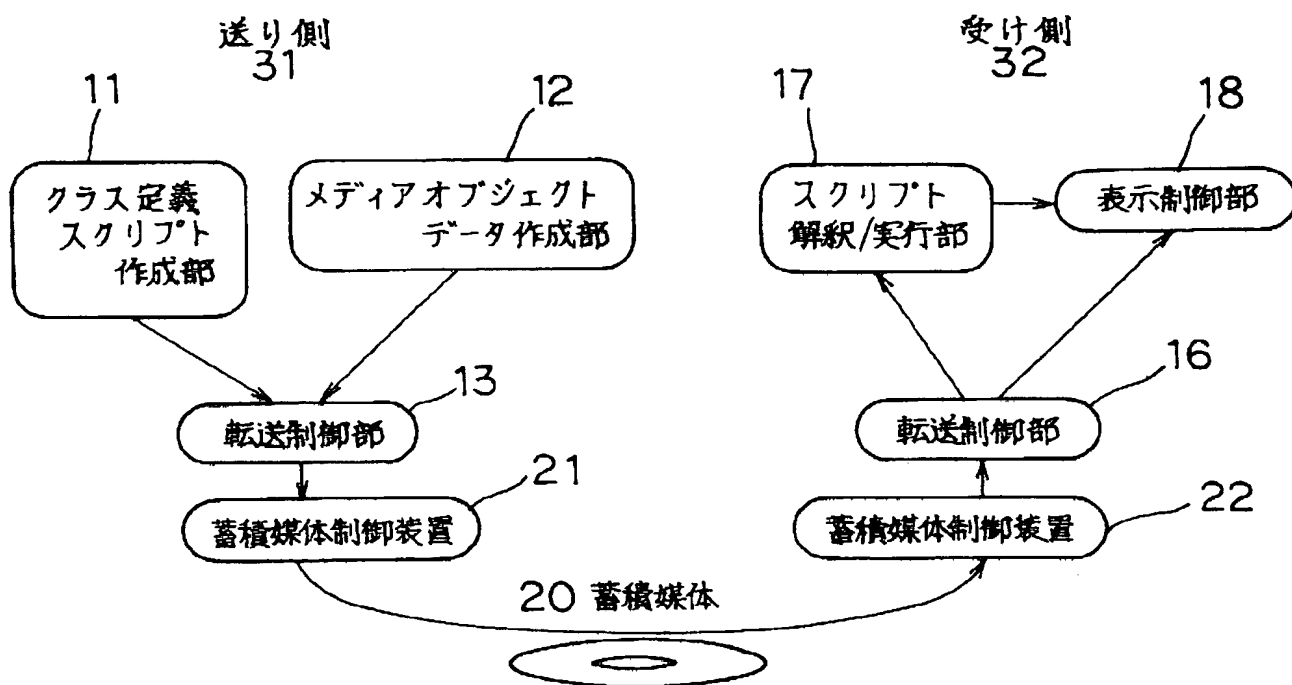
具体例1のシステム構成

【図6】



スクリプトの解釈/実行

【図7】



具体例2のシステム構成